PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-055835
(43)Date of publication of application : 20.02.2002

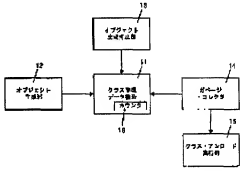
(51)Int.Cl. G06F 9/54
G06F 9/44
G06F 12/02

(21)Application number : 2000-244856
(22)Date of filing : 11.08.2000

(71)Applicant : OMRON CORP
(72)Inventor : KONAKA YOSHIHARU
HIRONO MITSUAKI
SAIKI HIDEAKI
KADOWAKI MASANORI
MURAI KENICHI

(54) PROGRAM UNLOADING SYSTEM AND STORAGE MEDIUM

(57)Abstract:
PROBLEM TO BE SOLVED: To clearly unload an optional program in each prescribed unit in a program execution mechanism having structure capable of loading a necessary program in each prescribed unit such as a class in accordance with the execution of the program.
SOLUTION: When class deletion is requested, an object generation suppression part 13 sets up an object generation suppression flag in class management data structure 11 to an inhibited state. When the flag is in the inhibited state, an object generation part 12 does not newly generate an object even when object generation is requested. Thus, objects are only recovered by a garbage collector 14 and gradually reduced. At the time of detecting the deletion of all objects, a class unload execution part 15 unloads all classes. Then the collector 14 deletes a class loader object for loading the classes.



LEGAL STATUS

[Date of request for examination] 27.11.2003
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision

of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2002－55835
（P2002－55835A）
(43)公開日　平成14年２月20日(2002.2.20)

(51)Int.Cl.⁷		識別記号	Ｆ Ｉ			ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ	9/54		Ｇ０６Ｆ	9/44	５３０Ｐ	５Ｂ０６０
	9/44	５３０		12/02	５３０Ｅ	５Ｂ０７６
	12/02	５３０		9/06	６４０Ｃ	

審査請求　未請求　請求項の数２　ＯＬ　（全 10 頁）

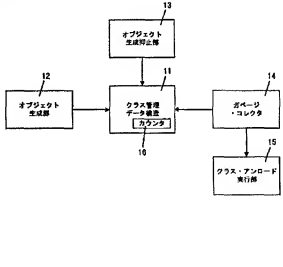
(21)出願番号　特願2000－244856(P2000－244856)
(22)出願日　平成12年８月11日(2000.8.11)

(71)出願人　000002945
オムロン株式会社
京都市下京区塩小路通堀川東入南不動堂町801番地
(72)発明者　小中　義治
京都府京都市右京区花園土堂町10番地　オムロン株式会社内
(72)発明者　廣野　光明
京都府京都市右京区花園土堂町10番地　オムロン株式会社内
(74)代理人　100094019
弁理士　中野　雅房

最終頁に続く

(54)【発明の名称】　プログラムアンロードシステム及び記憶媒体

(57)【要約】
【課題】　プログラム実行に応じて必要なプログラムをクラスなど所定単位でロードできる仕組みを有するプログラム実行機構において、任意のプログラムを所定単位で明示的にアンロードすることができるようにする。
【解決手段】　クラスを削除する要求があると、オブジェクト生成抑止部１３は、クラス管理データ構造１１内のオブジェクト生成抑止フラグを禁止状態にする。オブジェクト生成抑止フラグが禁止状態であると、オブジェクト生成部１２はオブジェクト生成を要求しても新たにオブジェクトを生成しない。このため、オブジェクトはガベージ・コレクタ１４によって回収されるだけとなり、次第に減少する。クラス・アンロード実行部１５は、全てのオブジェクトが削除されたことを検出すると、全てのクラスをアンロードする。続けて、ガベージ・コレクタ１４は、クラスをロードしていたクラスローダー・オブジェクトを削除する。



1

【特許請求の範囲】

【請求項１】　プログラム実行に応じて、必要なプログラムを所定単位でロードできる仕組みを持つプログラム実行機構において、

所定単位のプログラムから生成されるオブジェクトの生成を抑止する機構と、所定条件下でオブジェクトを消滅させる機構と、

所定単位のプログラムから生成されたオブジェクトが、前記オブジェクト消滅機構によって全て消滅させられたことを検出する機構と、

すべてのオブジェクトが消滅させられた所定単位のプログラムをシステムからアンロードする機構とを有することを特徴とするプログラムアンロードシステム。

【請求項２】　プログラム実行に応じて、必要なプログラムを所定単位でロードできる仕組みを持つプログラム実行機構において、

所定単位のプログラムから生成されるオブジェクトの生成を抑止する機構と、所定条件下でオブジェクトを消滅させる機構と、

所定単位のプログラムから生成されたオブジェクトが、前記オブジェクト消滅機構によって全て消滅させられたことを検出する機構と、

すべてのオブジェクトが消滅させられた所定単位のプログラムをシステムからアンロードする機構とを有するプログラムアンロードシステムを格納した記憶媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、プログラムアンロードシステムおよび記憶媒体に関する。特に、Ｊａｖａプログラム実行環境のようなプログラム実行機構で、クラスをアンロードする処理において、明示的にクラスをアンロードできるようにするためのものである。

【０００２】

【背景技術】プログラム実行に応じて必要なプログラムをクラス単位などでロードできる仕組みを持つプログラム実行機構、例えばＪａｖａプログラム実行環境（Ｊａｖａ言語で記述されたソースプログラムをコンパイルすることにより生成される、中間コードを実行するＪａｖａ仮想マシンによって実現されるプログラム実行環境）では、プログラムをインターネットやサーバ等からダウンロードして実行できるという特徴を持つ。この特徴を利用すれば、新たなプログラムをダウンロードしてシステム（ＪａｖａプログラムＪａｖａ実行環境搭載装置）に実装させることにより、システムを停止させることなく、システムの振る舞いや機能を変更することも可能になる。例えば、図１に示すように、不具合修正版のクラスＡ、機能追加（削除）版のクラスＡ、収集データ変更版などの新たなクラスＡをシステム１にロードすることにより、不具合のあるクラスＡを修正済みのクラスに置き換えたり、新たな機能を追加したり、不要な機能を除去したり、エージェントにより収集されたデータを変更または追加することができる。

2

【０００３】このような方式を実現するためには、システム動作中に変更または修正されたクラスを再びシステムにロードする必要がある。現在のＪａｖａプログラム実行環境では、同名のクラスを再ロードするためには、置き換えたいクラスを一旦システムから削除しなくてはならない。このようにクラスをシステムから削除することを「クラスをアンロードする」という。また、クラスとは、オブジェクト（１つのデータ構造で、データの値とそれに関する手続を合わせたものであって、クラスにより生成される。）内部のデータ構造とその操作を まとめて定義したものである。

【０００４】通常のＪａｖａプログラム実行環境においても、ロードされたクラスを再びアンロードする仕組みは組み込まれているが、通常のＪａｖａプログラム実行環境に組み込まれているクラスをアンロードする仕組みは、次に説明するように、あるクラスから生成されたオブジェクトが全てガベージ・コレクトの対象となったとき、そのクラスをアンロードするというものである。

【０００５】図２、図３及び図４は、Ｊａｖａプログラム実行環境において、Ｊａｖａ仮想マシンにクラスがロードされ、またアンロードされる仕組みを表している。例えば図２に示すクラスＡ１のオブジェクトが、その処理の中でクラスＡ２を呼び出すと、クラスＡ２がＪａｖａ仮想マシンにロードされ、図５（ａ）のステップＳ１に示すようにオブジェクト生成部がオブジェクト３の作成を要求するので、ロードされたクラスＡ２はＪａｖａ仮想マシン上でインスタンス化（Ｊａｖａ仮想マシン上で実行可能な状態となること）されてオブジェクト３（インスタンスともいう。）が作成される（ステップＳ２）。そして、呼び出したクラスＡ１のオブジェクト３は、クラスＡ２のオブジェクト３を利用できるようになる。このようにして、Ｊａｖａ仮想マシン上においては、図２に示すようにクラスローダー・オブジェクト２によって各クラスＡ１、Ａ２、…がロードされ、さらに各クラスＡ１、Ａ２、…からインスタンス化されたオブジェクト３の群が生成される。一方、ＪａｖａプログラムＪａｖａ実行環境においては、あるオブジェクト３が他のオブジェクト３によって参照もしくは使用されていないことがガベージ・コレクタによって確認されると（図５（ｂ）のステップＳ１１でYESの場合）、ガベージ・コレクタは使われなくなったオブジェクト３を回収（クリア）する（ステップＳ１２）。ついで、クラスＡ１、Ａ２、…から生成されたオブジェクト３が全てガベージ・コレクタによって回収されたか否か判別し（ステップＳ１３）、図３に示すようにクロスローダ・オブジェクト２から派生した全てのオブジェクト３が全て回収されていれば、クラスＡ１、Ａ２、…とクラスローダー・オブジェクト２をＪａｖａ仮想マシンからアンロードする

（ステップＳ１４）。なお、クラスローダーとは、ＪａｖａプログラムＪ実行環境において、クラスファイルを読み込み、クラスをメモリ上に展開するものであって、プログラム実行に必要な情報が納められている。

【０００６】しかし、オブジェクト３は任意のタイミングで生成されているため、従来のガベージ・コレクタによるオブジェクト回収方式では、使用されないオブジェクト３がガベージ・コレクタによって回収される一方、図４に示すように、クラスＡ１、Ａ２、…からオブジェクト３が生成され続けており、いつまでたっても全てのオブジェクト３が回収されることが無く、クラスＡ１、Ａ２、…をアンロードすることができない可能性がある。このため、いつまでたってもクラスを再ロードして置き換えることができない恐れがある。

【０００７】これを解決するためには、クラスに別名を付けて、ロードするという方法もある。例えば、既に組み込まれているクラスＡ ver1に対してクラスＡ ver1で置き換える代わりに、図６に示すように、クラスＡ ver2、クラスＡ ver3というように別名でロードし、別名でロードしたクラスＡ ver3を実行させるというものである。

【０００８】しかし、このような方法では、システム内に不要なクラス（上記の例では、クラスＡ ver1やクラスＡ ver2）が残ってしまい、メモリの使用効率が悪くなる。パソコン上のＪａｖａプログラム実行環境では、図６（ａ）に示すように、使用するメモリ領域を拡張することができるので、メモリの使用効率についてはそれほど注意を払う必要はなく、クラスＡ ver1やクラスＡ ver2などの古いクラスを残しておいても差し支えない。これに対し、組込み機器上のＪａｖａプログラム実行環境では、図６（ｂ）に示すように、使用できるメモリに厳しい制約があるため、クラスＡ ver1やクラスＡ ver2などの古いクラスをいつまでもメモリ上に置いてメモリを無駄に使用することは許されない。また、パソコン上のＪａｖａプログラム実行環境でも、古いクラスは削除してメモリを効率的に使用することが望ましい。

【０００９】

【発明の開示】本発明の目的とするところは、プログラム実行に応じて必要なプログラムをクラスなど所定単位でロードできる仕組みを有するプログラム実行機構において、任意のプログラムを所定単位で明示的にアンロードすることができるようにすることにある。

【００１０】本発明にかかるプログラムアンロードシステムは、プログラム実行に応じて、必要なプログラムを所定単位でロードできる仕組みを持つプログラム実行機構において、所定単位のプログラムから生成されるオブジェクトの生成を抑止する機構と、所定条件下でオブジェクトを消滅させる機構と、所定単位のプログラムから生成されたオブジェクトが、前記オブジェクト消滅機構によって全て消滅させられたことを検出する機構と、すべてのオブジェクトが消滅させられた所定単位のプログラムをシステムから削除する機構とを有することを特徴としている。

【００１１】また、本発明にかかる記憶媒体は、プログラム実行に応じて、必要なプログラムを所定単位でロードできる仕組みを持つプログラム実行機構において、所定単位のプログラムから生成されるオブジェクトの生成を抑止する機構と、所定条件下でオブジェクトを消滅させる機構と、所定単位のプログラムから生成されたオブジェクトが、前記オブジェクト消滅機構によって全て消滅させられたことを検出する機構と、すべてのオブジェクトが消滅させられた所定単位のプログラムをシステムから削除する機構とを有するプログラムアンロードシステムを格納したものである。

【００１２】ここで、プログラムをロードする所定単位とは、例えばＪａｖａ仮想マシンにおけるクラスであるが、これに限るものではない。また、オブジェクトを消滅させる機構とは、例えばＪａｖａ仮想マシンにおけるガベージ・コレクタであるが、これに限るものでなく、オブジェクトを消滅させる所定条件とは、例えば当該オブジェクトが使用されないことである。また、プログラムアンロードシステムを格納する記憶媒体の種類は、特に限定されるものではないが、例えばハードディスク、ＣＤ類、ＤＶＤ、各種ＩＣメモリ、ＭＯなどが含まれる。

【００１３】本発明にあっては、クラス等の所定プログラムから生成されるオブジェクトの生成を抑止する機構を備えているので、目的とするプログラムのオブジェクト生成を抑止すると、オブジェクト消滅機構によって次第にオブジェクトの数が減少させられていく。そして、所定プログラムから生成した全てのオブジェクトが消滅したことを前記検出機構により検出すると、当該プログラムをシステムから削除する。よって、本発明によれば、目的とするプログラムのオブジェクト生成を抑止させることで、明示的に目的とするプログラムを削除することができる。

【００１４】

【発明の実施の形態】（第１の実施形態）図７は本発明の一実施形態によるプログラムアンロードシステムの構成を示す概略図である。このプログラムアンロードシステムは、クラス管理データ構造１１、オブジェクト生成部１２、オブジェクト生成抑止部１３、ガベージ・コレクタ１４、クラス・アンロード実行部１５から構成されている。クラス管理データ構造１１は、クラス及びオブジェクトを管理しており、クラスローダー・オブジェクトによってロードされた各クラスから生成されているオブジェクトの数をカウンタに保持し更新している。また、クラス管理データ構造１１は、各クラスのオブジェクトの生成を許可又は禁止するためのフラグ（以下、オブジェクト生成抑止フラグという。）を保持している。

なお、このプログラムアンロードシステムの各構成要素は、主としてソフトウエアによって構成されるものであり、例えば、ハードディスク、ＣＤ－ＲＯＭやＣＤ－ＲＷ等のＣＤ類、ＤＶＤ、ＭＯなどの記憶媒体に格納される。

【００１５】クラス管理データ構造１１のオブジェクト生成抑止フラグは、通常は許可状態となっているが、オブジェクト生成抑止部１３は、要求によりクラス管理データ構造１１のオブジェクト生成抑止フラグを禁止状態に設定することができる。

【００１６】オブジェクト生成部１２は、要求に応じて各クラスからオブジェクトを生成させるものであって、オブジェクト生成抑止フラグが許可状態の場合には、要求に応じてオブジェクトを生成し、クラス管理データ構造１１内に保持されている、オブジェクト数カウント用のカウンタ１６を１増加させるが、オブジェクト生成抑止フラグが禁止状態の場合には、オブジェクト生成を要求されてもオブジェクトを生成しない。すなわち、図８のフロー図に示すように、オブジェクト作成要求があったとき（ステップＳ２１）には、オブジェクト生成部１２は、オブジェクト生成抑止フラグを参照して当該フラグが許可状態か禁止状態か判定する（ステップＳ２２）。そして、クラス管理データ構造１１のオブジェクト生成抑止フラグが許可状態ならば（ステップＳ２２でＮｏの場合）、クラスからオブジェクトを生成し（ステップＳ２４）、クラス管理データ構造１１に保持しているオブジェクト数のカウンタ１６を１増加させる（ステップＳ２５）。オブジェクト生成抑止フラグが禁止状態ならば（ステップＳ２２でＹＥＳの場合）、オブジェクト生成要求があってもオブジェクト生成部１２は新たなオブジェクトの生成を禁止され、新たなオブジェクトを生成しない（ステップＳ２３）。

【００１７】ガベージ・コレクタ１４は、使用されなくなったオブジェクトに割り当てられていたメモリ領域を自動的に解放するメモリ管理機構である。このガベージ・コレクタ１４は、図９のフロー図に示すように、オブジェクトの使用状況を監視しており（ステップＳ３１）、いずれかのオブジェクトが使用されていない場合には、当該オブジェクトを回収し（ステップＳ３２）、クラス管理データ構造１１に保持しているオブジェクト数のカウンタ１６を１減少させる（ステップＳ３３）。また、派生している全てのクラスがオブジェクトを持たなくなったクラスローダー・オブジェクトは、ガベージ・コレクタ１４による回収の対象となる。

【００１８】クラス・アンロード実行部１５は、クラス管理データ構造１１に保持しているオブジェクト数を参照し、オブジェクトを生成していないクラスをアンロードするものである。すなわち、図１０のフロー図に示すように、クラスアンロードの実行を開始する指示があった場合には（ステップＳ４１）、オブジェクト生成抑止部１３は、クラス管理データ構造１１内のオブジェクト生成抑止フラグを禁止状態に設定する（ステップＳ４２）。一方、クラス・アンロード実行部１５は、クラスローダー・オブジェクトによって生成された全てのクラスのオブジェクトを監視しており（ステップＳ４３）、全てのクラスのオブジェクトが回収されてクラス管理データ構造１１内のオブジェクト数のカウンタ１６がゼロになると、全てのクラスをシステムからアンロードする（ステップＳ４４）。これによって、クラスローダー・オブジェクトによって読み込まれ、メモリ上に展開されたクラスが削除される。

【００１９】図１１（ａ）（ｂ）及び図１２（ｃ）（ｄ）（ｅ）は、上記のような構成のプログラムアンロードシステムにより、クラスＡ１、Ａ２、…及びオブジェクト３が生成されてから、完全に削除されるまでの様子を模式的に表している。アプリケーションプログラムが実行されると、図１１（ａ）に示すように、Ｊａｖａ仮想マシン内にクラスローダー・オブジェクト２が生成され、クラスローダー・オブジェクト２によってプログラム実行に必要なクラスＡ１、Ａ２、…がロードされる。さらに、各クラスＡ１、Ａ２、…からは、オブジェクト生成部１２により任意にオブジェクト３が生成される。一方、使用されなくなったオブジェクト３は、ガベージ・コレクタ１４によって回収されている。従って、クラス管理データ構造１１内のオブジェクト生成抑止フラグが許可状態になっている場合には、常にオブジェクト３は生成と消滅を繰り返している。

【００２０】これに対し、例えば新規のクラスをロードするために実行中のクラスを削除する必要が生じた場合には、アンロード指示を受けてオブジェクト生成抑止部１３がオブジェクト生成抑止フラグを禁止状態に設定する。当該フラグが禁止状態になると、図１１（ｂ）に示すように、それ以降オブジェクト生成部１２によって各クラスＡ１、Ａ２、…からオブジェクト３が生成されなくなるので、使用されないオブジェクト３がガベージ・コレクタ１４によって回収されることにより、次第にオブジェクト数は減少していく。

【００２１】クラスローダー・オブジェクト２によりロードされた全てのクラスＡ１、Ａ２、…から生成しているオブジェクト３がすべてガベージ・コレクタ１４によって回収され、図１２（ｃ）のようにクラスＡ１、Ａ２、…から生成したオブジェクト３が存在しなくなると、クラスローダー・オブジェクト２もガベージ・コレクタ１４による回収の対象となる。オブジェクト３がすべてガベージ・コレクタ１４によって回収されたかどうかは、クラス管理データ構造１１に設けられているオブジェクト数のカウンタ１６がゼロになることで判断される。クラスローダー・オブジェクト２がガベージ・コレクタ１４による回収の対象となり、カウンタ１６がゼロになると、図１２（ｃ）に示すように、クラス・アンロ

7

ード実行部１５によって全てのクラスＡ１、Ａ２、…のアンロードが開始され、クラスＡ１、Ａ２、…がすべて削除されると、クラスローダー・オブジェクト２もガベージ・コレクタ１４によって削除される。

【００２２】この結果、Ｊａｖａ仮想マシンに新しいクラスをロードすることができるようになる。

【００２３】（第２の実施形態）上記実施形態では、クラスローダー・オブジェクト２によってロードされた全てのクラスから生成されたオブジェクト数をクラス管理データ構造１１のカウンタ１６で数えるようにしていたが、オブジェクト数を数える代わりに、生成されているオブジェクトのリストをクラス管理データ構造１１内に保持させるようにしてもよい。すなわち、この実施形態では、図１３に示すように、オブジェクト生成部１２でオブジェクトを生成させた場合には、当該オブジェクトをクラス管理データ構造１１のリストに追加する（ステップＳ２５ａ）。また、ガベージ・コレクタ１４によってオブジェクトを削除した場合には、クラス管理データ構造１１のリストから当該オブジェクトを削除する（ステップＳ３３ａ）。また、リストにオブジェクトが登録されていないと判断したとき（ステップＳ４３ａ）、クラス・アンロード実行部１５によって全てのクラスをシステムから削除する（ステップＳ４４ａ）。

【００２４】このような実施形態でも、第１の実施形態と同様にしてクラスを明示的にアンロードすることが可能になる。

【００２５】

【発明の効果】本発明によれば、目的とするプログラムのオブジェクト生成を抑止させることで、明示的に目的とするプログラムを削除することができる。よって、確実にシステムに新たなプログラムを再ロードすることができる。

【図面の簡単な説明】

【図１】システムにロードされているクラスを再ロードする必要のある状況を説明する図である。

【図２】従来のＪａｖａプログラミング実行環境において、クラスローダー・オブジェクトによりクラスがロードされ、各クラスからオブジェクトが生成している様子を説明する図である。

【図３】同上のＪａｖａプログラミング実行環境において、全てのオブジェクトが回収（削除）された状態を説明する図である。

【図４】同上のＪａｖａプログラミング実行環境において、新しいオブジェクトが生成している様子を説明する図である。

8

【図５】（ａ）は同上のＪａｖａプログラミング実行環境における、オブジェクト生成部の働きを示すフロー図、（ｂ）はガベージ・コレクタの働きを示すフロー図である。

【図６】（ａ）はクラスをロードされたパソコン上のＪａｖａプログラミング実行環境におけるメモリ領域使用状況を示す図、（ｂ）はクラスをロードされた組込み機器上のＪａｖａプログラミング実行環境におけるメモリ領域使用状況を示す図である。

【図７】本発明の一実施形態によるプログラムアンロードシステムの構成を示す図である。

【図８】同上のプログラムアンロードシステムに用いられているオブジェクト生成部の動作を説明するためのフロー図である。

【図９】図７のプログラムアンロードシステムに用いられているガベージ・コレクタの動作を説明するためのフロー図である。

【図１０】図７のプログラムアンロードシステムに用いられているオブジェクト生成抑止部とクラス・アンロード実行部の動作を説明するためのフロー図である。

【図１１】（ａ）（ｂ）は、クラスがロードされ、各クラスからオブジェクトが生成してからクラス及びクラスローダー・オブジェクトが削除されるまでの過程を説明する図である。

【図１２】（ｃ）、（ｄ）及び（ｅ）は、図１１（ａ）（ｂ）の続図である。

【図１３】本発明の別な実施形態によるプログラムアンロードシステムに用いられているオブジェクト生成部の動作を説明するためのフロー図である。

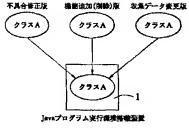
【図１４】同上の実施形態によるプログラムアンロードシステムに用いられているガベージ・コレクタの動作を説明するためのフロー図である。

【図１５】同上の実施形態によるプログラムアンロードシステムに用いられているオブジェクト生成抑止部とクラス・アンロード実行部の動作を説明するためのフロー図である。

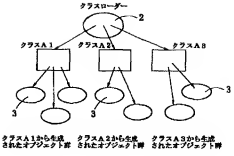
【符号の説明】

２　クラスローダー・オブジェクト
３　オブジェクト
Ａ、Ａ１、Ａ２、…　クラス
１１　クラス管理データ構造
１２　オブジェクト生成部
１３　オブジェクト生成抑止部
１４　ガベージ・コレクタ
１５　クラス・アンロード実行部

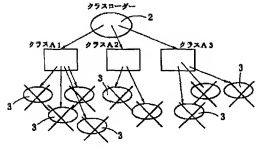
【図１】



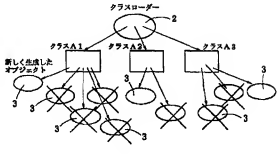
【図２】



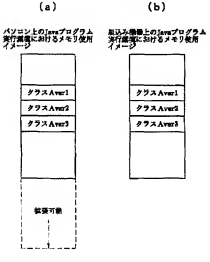
【図３】



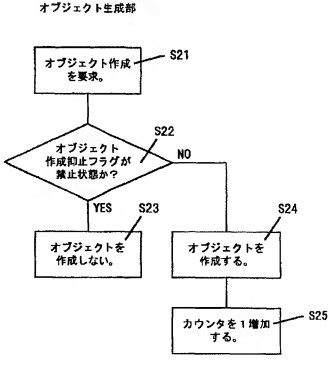
【図４】



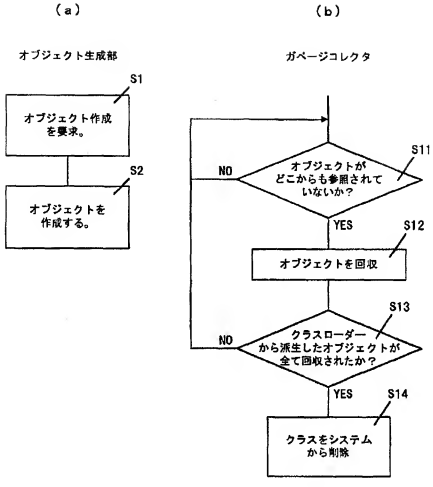
【図６】



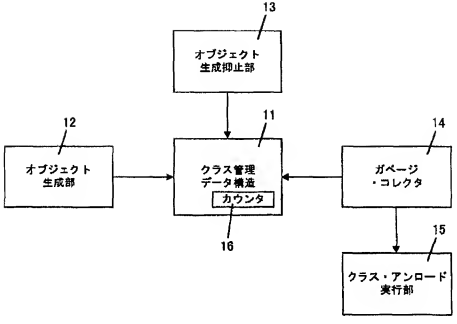
【図８】



【図５】

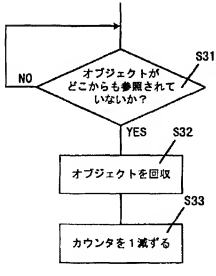

（ａ）
（ｂ）
オブジェクト生成部
ガベージコレクタ
S1
オブジェクト作成を要求。
S2
オブジェクトを作成する。
S11
NO
オブジェクトがどこからも参照されていないか？
YES
S12
オブジェクトを回収
S13
NO
クラスローダーから派生したオブジェクトが全て回収されたか？
YES
S14
クラスをシステムから削除

【図７】



【図９】

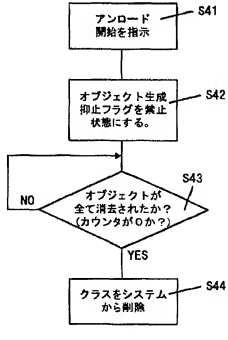
ガベージ・コレクタ



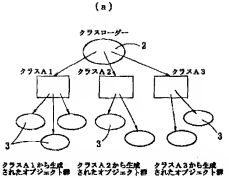
【図１０】

オブジェクト生成抑止部

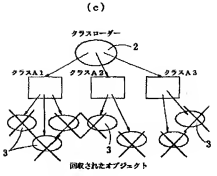
クラス・アンロード実行部



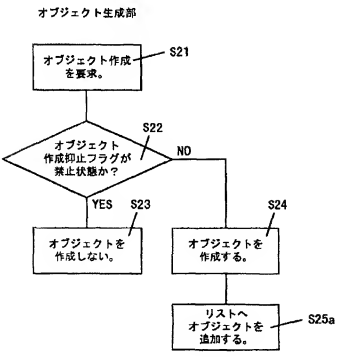
【図１１】



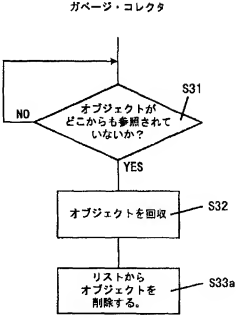
【図１２】



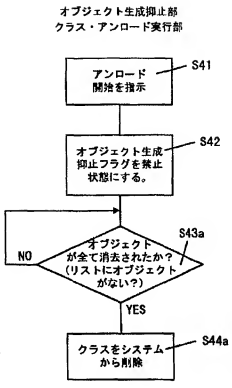
【図１３】



【図１４】



【図１５】



フロントページの続き

(72)発明者　齋木　秀明
京都府京都市下京区西洞院木津屋橋通東入ル　オムロンソフトウエア株式会社内
(72)発明者　門脇　正規
京都府京都市右京区花園土堂町10番地　オムロン株式会社内
(72)発明者　村井　謙一
京都府京都市右京区花園土堂町10番地　オムロン株式会社内

Ｆターム(参考)　5B060 AA09 AA18 AC01
5B076 BA01 BA05